

Anwenderdoku zur Ermittlung der Kabellängen

- Installation des Programmes und Schnellstart-Guide
 - Installation
 - Schnellstart und Verwendung des Programms
 - Standardmäßige Ausgabe des Programms
 - .dxf-Datei: *dxfdtei_cables.dxf*
 - Excel-Datei: *dxfdtei_cables.xlsx*
 - Excel-Datei: *dxfdtei_BOM.xlsx*
- Fehlerfälle und Fehlerbehandlung
 - Übersicht
 - Zwei Haupt-Workflows
 - Fehler- und Warnungskategorien
 - Fehlerausgabe
 - Allgemeine Fehler und Warnungen (beide Workflows)
 - Struktur der Error-JSON-Datei
 - Arbeiten mit Fehlerfällen
 - Beispiele aus Testdaten
- Anpassung des Verhaltens des Programms:
 - Verwendete Konfigurationsdateien
 - Zusammenfassung und Struktur der Konfigurationsdateien
 - *allgemein.cfg*
 - *BMK.cfg*
 - *kabel.cfg*
 - *bezeichner.cfg*
 - Validierung der Konfigurationsdateien
- Anwenderhinweise zur Layouerstellung
 - Verwendung der Layer
 - Zeichnen von Kabeltrassen
 - Attribute der Blöcke (Bauteile)
 - Positionierung von Bauteilen

Installation des Programmes und Schnellstart-Guide

Installation

Das Programm mit allen Quellen befindet sich auf dem hauseigenen git Server und kann mit einen beliebigen git client auf einem Anwenderrechner über `git clone`

<http://gitea.schoenenberger.de/mistangl/kabellaengen.git> geholt werden. Dieser Ordner kann auch einfach mit allen Unterordnern gezippt und auf einem anderen Rechner entpackt werden.

Danach muss auf dem Zielrechner (z.B. per Windows APP) Python 3.X installiert werden. Alternativ kann ein Python Interpreter von einem Netzlaufwerk verwendet werden. Der Pfad zu diesem Interpreter muss dann über eine *Umgebungsvariable* mit dem Namen **NETWORK_INTERPRETER_PATH** auf der Maschine gesetzt

sein. Grundsätzlich wird eine lokale Installation eines Python Interpreters (von <https://www.python.org/downloads/>) empfohlen.

Zur Installation des Kabeltools muss in dem geklonten Ordner `kabellaengen\bin` die Datei `install.bat` als **Administrator** ausgeführt werden. Hierfür *Rechtsklick* auf die Datei und *als Administrator ausführen*. Es wird in der Folge ein Ordner auf dem Desktop des Computers mit dem namen *Kabeltool* erstellt. In diesem Ordner ist eine Verknüpfung `create_cables`.

Ist Python installiert, werden per Doppelklick automatisch auf die `bin\install_py.bat` alle nötigen python Package heruntergeladen. Diese werden dann automatisch aus dem Netz in den `.venv` kopiert.

Schnellstart und Vewendung des Programms

Im auf dem Desktop erzeugten Ordner findet sich das eigentliche Programm `create_cables`. Um das Programm auszuführen, genügt es die zu verarbeitende **.dxf-Datei** per *Drag and Drop* auf das Programm zu ziehen. Das öffnen der *Command-Shell* bestätigt den Start des Programms und der Anwender wird über den Ablauf informiert.

Die Ausgabe schreibt:

```
=== Fetching Positions ===
Given .dxf-file is ASCII-dxf. Proceeding to use iterative functions. Process may
take longer.

Validating given configs: Checking for inconsistency.
No Inconsistencies found. Continuing with routing process.

writing results to a json file ...
done

=== Creating Graph for Routing ===
writing results to a json file ...
done

=== Writing Output Files ===
Cable-Routes exported to new dxf-file
Cable-Summary exported to Excel-file
BOM exported to Excel-file

C:\10-Develop\kabellaengen\work\easy_BOM.xlsx
C:\10-Develop\kabellaengen\work\easy_cables.dxf
C:\10-Develop\kabellaengen\work\easy_cables.xlsx
C:\10-Develop\kabellaengen\work\easy_positions.json
C:\10-Develop\kabellaengen\work\easy_todraw.json
      5 Datei(en) verschoben.
Drücken Sie eine beliebige Taste . . .
```

Das Drücken einer beliebigen Taste beendet das Programm und die Ausgabe kann verwendet werden.

[!IMPORTANT] Der Dateiname darf **keine Leerzeichen oder Sonderzeichen** enthalten. Verwende z. B. projekt_123.dxf statt mein projekt #1.dxf.

Standardmäßige Ausgabe des Programms

In den Ordner auf dem Desktop in dem das Kabeltool liegt werden nach Beendigung der Berechnungen die Ausgabedateien abgelegt. Standardmäßig sind dies vier Dateien, wovon letztlich zwei für den Anwender bestimmt sind. In der nachfolgenden Liste steht der name *dxfdtei* stellvertretend für die eingegebene Datei des Benutzers.

Dateiname	Details
<i>dxfdtei_cables.dxf</i>	.dxf-Datei, mit Kabelwegen (z. B. für Import ins Original-Layout als eigenes Layer) und gezeichneten Racks (ggf. in 3D)
<i>dxfdtei_cables.xlsx</i>	Excel-Datei mit Kabellängen, Längenübersicht und Fehler-Übersicht
<i>dxfdtei_BOM.xlsx</i>	Excel-Datei mit Gesamtstückliste und Unterverteiler-spezifischer Stückliste
<i>dxfdtei_positions</i>	Zwischendatei (nicht für Endnutzer bestimmt)
<i>dxfdtei_todraw</i>	Zwischendatei (nicht für Endnutzer bestimmt)

.dxf-Datei: *dxfdtei_cables.dxf*

Die standardmäßig erstellte und abgelegte Ausgabedatei *dxfdtei_cables.dxf* enthält die Kabelwege aller während des Programmablaufs behandelten Kabelverbindungen und kann beispielsweise als neues Layer in die .dxf-Datei der Anlage importiert werden. Neben den Kabelwegen werden auch die Kurzbezeichnungen (z.b. MA0060, BG3240, ...) sowie die Kürzel der Unterverteiler auf jeweils separaten Layern in die Zeichnungsdatei exportiert und ermöglichen damit die in sich schlüssige Verwendung dieser Datei. Desweiteren werden die von dem Programm erkannten Kabeltrassen eingezeichnet und nummeriert. Sofern diese im Ursprungslayout dreidimensional angelegt wurden, wird auch das in der Ausgabe berücksichtigt.

Excel-Datei: *dxfdtei_cables.xlsx*

Die standardmäßig erstellte und abgelegte Ausgabedatei *dxfdtei_cables.xlsx* enthält im fehlerfreien Fall drei Worksheets:

Length by ID:

zeigt die Kabel-ID, die tatsächliche Länge des Kabels, die Kabel-Atikelnnummer und zuletzt den zugehörigen (aber gekürzten Bezeicher) für das Kabel:

Cable-ID	True Length (m)	Cable-ArtNr	Cable-Name (short)
UC0101-BG3241	2,9213	722001332	M12 St-0°/ M12 Bu-90° 3m
UC0101-BG3240	7,1784	722001334	M12 St-0°/ M12 Bu-90° 10m
UC0101-BG3270	15,4686	722001336	M12 St-0°/ M12 Bu-90° 20m
UC0101-BG3260	27,4471	722001339	M12 St-0°/ M12 Bu-90° 30m
UH01-MA0062	10,3015	725000015	4G1,5mm ² , Steuerleitung

Cables SIVAS:

zeigt die SIVAS-Nummer der Kabel und daneben die benötigte Anzahl an Kabeln in diesem Layout bzw. (für den Fall von Kabeln als Meterware, z.b. bei Motoren) die kummulierte benötigte Länge:

Cable-ArtNr	Amount (pcs)	Cumm. Length (m)
722001332	1	
722001334	1	
722001336	1	
722001339	1	
725000015		11

Rack-Lengths:

zeigt die Längen der einzelnen Kabeltrassen-Segmente anhand deren Nummerierung innerhalb der dxf Datei. Diese Ausgabe dient der Überprüfung der benötigten Gesamtlänge der Kabeltrassen.

Rack-ID	Length (m)
Rack_1	6,5
Rack_2-1	3,5
Rack_2-2	15

Erweiterte Ausgabe im Fehlerfall:

Für den Fall, dass während des Programmablaufs Fehler auftreten bzw. Fehler im Layout erkannt werden, werden diese in weitere Worksheets geschrieben. Diese Worksheets tragen das Präfix *ERR*

- ERR-Equipment-Connection:
 - Zeigt alle nicht mit den Kabeltrassen verbundenen Elemente (Sensoren / Aktoren / Unterverteiler) an
 - Gibt neben dem Typen, den Bezeichner und die x, y-Koordinate des Elements zurück
 - Verbindungsfehler möglicherweise aufgrund zu großen Abstandes zur nächsten Kabeltrasse
- ERRORS-Routing - Zeigt alle fehlgeschlagenen Kabelverbindungen an
 - Fallunterscheidung:
 - Unterverteiler nicht in Layout vorhanden: Unterverteiler ist in der Kennzeichnung der Sensor-Blöcke im Layout erwähnt aber nicht im Layout selbst positioniert. Es wird nur der betroffene Unterverteiler angezeigt. Keine explizite Auflistung aller damit ebenfalls nicht verbundenen Sensoren / Aktoren
 - Unterverteiler **und** Sensor / Aktor nicht angebunden: Sowohl Sensor als auch Aktor sind nicht mit den Kabeltrassen verbunden. Beide damit auch in *ERR-Equipment-Connction* aufgeführt
 - Unterverteiler nicht angebunden: siehe oben.
 - Sensor / Aktor nicht angebunden siehe oben.
- ERR-Attributes:
 - Zeigt alle Blöcke an, welche Fehler in deren Attributen aufweisen

- Mögliche Fehler:
 - Kein Attribut "KENNZEICHNUNG": keine Zuweisung zu Unterverteiler möglich. Element wird nicht verkabelt und nicht gezeichnet.
 - Ungültiger Pfad in "KENNZEICHNUNG": Eingegebenes Attribut in Kennzeichnung entspricht nicht dem normalen Muster. Element wird nicht verkabelt und nicht gezeichnet.

Excel-Datei: *dxfdtei_BOM.xlsx*

Diese Datei enthält eine vollständige **Stückliste (BOM)** aller im Layout erkannten Komponenten – also sowohl aller Sensoren/Aktoren als auch der Kabel, die zum Einsatz kommen. Die Datei besteht aus zwei Tabellenblättern:

1. BOM (Gesamtstückliste):

Zeigt für jede verwendete Artikelnummer den Namen laut *bezeichner.cfg*, die Gesamtanzahl bzw. Gesamtlänge. Sensoren (bzw. andere Komponenten mit Artikelnr) werden über ihre Artikelnummer gezählt. Kabel werden – sofern Meterware (z.B. Steuerleitungen für Motoren) – über die summierte Länge dargestellt, andernfalls über Stückzahl.

Art.-Number	Amount (pcs)	Length (m)	Name (SIVAS)
722001330	3		M12 Sensorleitung 1 m, gewinkelt
725000015		24	Steuerleitung MA, 4G1,5mm ²
839220147	5		Reflexionslichttaster XYZ

2. BOM by UV (Stückliste pro Unterverteiler):

Dieses Worksheet gibt für jede Artikelnummer aus, **welchem Unterverteiler (UV)** sie zugeordnet ist und wie viele davon jeweils dorthin gehören. Kabelmeter werden wie im anderen Arbeitsblatt getrennt ausgewiesen.

UV	Art.-Number	Amount (pcs)	Length (m)	Name (SIVAS)
UH01	722001330	1		M12 Sensorleitung 1 m, gewinkelt
UC01	722001330	2		M12 Sensorleitung 1 m, gewinkelt
UC01	725000015		12	Steuerleitung MA, 4G1,5mm ²

Hinweis: Diese Ansicht ist besonders nützlich zur **vorbereitenden Bestellplanung pro UV**.

Fehlerfälle und Fehlerbehandlung

Übersicht

Das Programm führt während der Verarbeitung umfassende Validierungen durch und erkennt verschiedene Arten von Fehlern im Layout. Die Fehlerbehandlung unterscheidet sich je nach Anwendungsfall:

Zwei Haupt-Workflows

Das Kabeltool unterstützt zwei verschiedene Workflows:

1. **Cable Routing Workflow** ([getexdraw.bat](#)):

- Schritt 1: [getpositions.py](#) - Extrahiert Positionen und validiert Layout
- Schritt 2: [routing.py](#) - Berechnet Kabelwege über Graph-Algorithmen
- Schritt 3: [drawdxf.py](#) - Erzeugt DXF mit Kabelwegen und Excel-Listen
- **Ausgabe:** [*_cables.dxf](#), [*_cables.xlsx](#), [*_BOM.xlsx](#)

2. **I/O Conversion Workflow** ([ioconverter.bat](#)):

- Schritt 1: [getpositions.py](#) - Extrahiert Positionen und validiert Layout
- Schritt 2: [portalexport.py](#) - Konvertiert zu Excel-Formaten (TIA, WSCAD, EA)
- **Ausgabe:** [*_TIA.xlsx](#), [*_WSCAD.xlsx](#), [*_EA.xlsx](#)

Fehler- und Warnungskategorien

Die Fehlerbehandlung unterscheidet zwischen zwei Kategorien:

- **Fehler (errors):** Schwerwiegende Probleme, die die Korrektheit der Ausgabe beeinträchtigen oder zum Abbruch führen
- **Warnungen (warnings):** Hinweise auf potenzielle Probleme, das Programm kann aber fortfahren

Fehlerausgabe

Für beide Workflows gilt:

- Falls Fehler oder Warnungen auftreten, schreibt [getpositions.py](#) **automatisch** eine JSON-Datei [*_errors.json](#) in den [work](#)-Ordner
- Im **Cable Routing Workflow** werden zusätzliche Fehler in der Excel-Datei [*_cables.xlsx](#) als separate Worksheets ausgegeben (siehe [ERR-Worksheets](#))

Im Ordner [testdata/](#) finden sich Beispiel-DXF-Dateien, die gezielt bestimmte Fehlerfälle demonstrieren (z.B. [easy_3tunnels_errors.json](#), [easy_tunnelerror1.json](#)).

Allgemeine Fehler und Warnungen (beide Workflows)

Die folgende Tabelle zeigt alle möglichen Fehler- und Warnungstypen, die von [getpositions.py](#) erkannt werden. Diese gelten **für beide Workflows** (Cable Routing und I/O Conversion):

Fehlertyp	Kategorie	Beschreibung	Ursache	Behebung
double_ids	Fehler	Mehrere Blöcke haben dieselbe ID (z.B. MA0062@1)	Doppelte Bauteile mit identischer Kennzeichnung und SPS-Präfix im Layout	IDs eindeutig vergeben oder doppelte Blöcke entfernen

Fehlertyp	Kategorie	Beschreibung	Ursache	Behebung
<code>missing_attributes</code>	Warnung	Block hat fehlende oder unvollständige Attribute	Block enthält z.B. nur A, B, C, ARTINR aber keine SPS-, IO- oder KENNZEICHNUNG-Attribute	Fehlende Attribute im DXF-Block ergänzen (z.B. SPS, KENNZEICHNUNG)
<code>missing_distributors</code>	Fehler	Unterverteiler in KENNZEICHNUNG erwähnt aber nicht im Layout	Sensor verweist auf UV (z.B. UH01) in KENNZEICHNUNG, aber UV-Symbol fehlt im Layout	Unterverteiler-Block im Layout positionieren oder KENNZEICHNUNG korrigieren
<code>missing_sensors</code>	Fehler	Sensor/Aktor nicht gefunden	Sensor wird in Mappings referenziert, aber Block fehlt oder hat ungültige Attribute	Sensor-Block im Layout ergänzen oder fehlerhafte Attribute korrigieren
<code>missing_tunnel</code>	Fehler	Tunnel hat weniger als 2 Positionen	Tunnel-Eingang oder -Ausgang fehlt (TUNNEL-Block oder MTEXT mit Muster <code>TUNNEL<nr>-<länge></code>)	Zweiten Tunnel-Eingang/Ausgang im Layout ergänzen
<code>overdefined_tunnel</code>	Fehler	Tunnel hat mehr als 2 Positionen	Mehr als zwei TUNNEL-Blöcke mit demselben Namen (z.B. TUNNEL2 dreimal)	Überzählige Tunnel-Positionen entfernen
<code>tunnel_missing_length</code>	Warnung	Tunnel-Block hat kein LAENGE-Attribut	LAENGE-Attribut fehlt in Tunnel-Block, Default-Länge (5m) wird verwendet	LAENGE-Attribut im Tunnel-Block ergänzen
<code>mapping_warnings</code> → <code>keine KENNZEICHNUNG</code> vorhanden	Warnung	KENNZEICHNUNG-Attribut fehlt komplett	Block hat kein KENNZEICHNUNG-Attribut	KENNZEICHNUNG-Attribut im Format <code>=<Anlage>+<UV>-<Karte></code> ergänzen

Fehlertyp	Kategorie	Beschreibung	Ursache	Behebung
<code>mapping_warnings</code> → Ungültiger Pfad in Kennzeichnung	Warnung	KENNZEICHNUNG hat falsches Format	Trennzeichen <code>+</code> , <code>-</code> oder <code>=</code> fehlen oder falsch positioniert	KENNZEICHNUNG-Format korrigieren (z.B. <code>=A01+UH01-KF1DQ04</code>)
<code>missing_sps_prefix</code>	Fehler	SPS-Präfix fehlt für Block-ID	Beim Zusammenführen von Blöcken fehlt der SPS-Präfix	SPS-Attribut in Block ergänzen

Für eine genaue Übersicht siehe in der normalen Anwenderdokumentation.

Struktur der Error-JSON-Datei

Die `*_errors.json`-Datei ist wie folgt strukturiert:

```
{
  "errors": {
    "missing_distributors": ["UH02", "UC05"],
    "overdefined_tunnel": ["TUNNEL2"],
    ...
  },
  "warnings": {
    "missing_attributes": {
      "BX0101": "Nur ein Block und/oder fehlende Attribute: dict_keys([...])"
    },
    "mapping_warnings": {
      "MA0099": "keine KENNZEICHNUNG vorhanden"
    },
    "tunnel_missing_length": {
      "TUNNEL1": "Tunnel 'TUNNEL1' hat keine LAENGE-Angabe. Verwende Default-Länge: 5m"
    }
  }
}
```

Jeder Fehlertyp enthält entweder:

- Eine **Liste** von betroffenen IDs/Namen (z.B. `["UH02", "UC05"]`)
- Ein **Dictionary** mit IDs als Keys und detaillierten Beschreibungen als Values

Arbeiten mit Fehlerfällen

Empfohlene Vorgehensweise:

1. **Programm ausführen** – Das Programm läuft auch bei Fehlern durch und erstellt soweit möglich Ausgaben
2. **Error-Datei prüfen** – Falls `*_errors.json` erstellt wurde, enthält sie alle erkannten Probleme

3. **Fehler beheben** – Layout gemäß Tabelle oben korrigieren
4. **Erneut ausführen** – Nach Korrektur sollte keine Error-Datei mehr erstellt werden

Hinweis: Die Error-Datei wird **nur dann** erstellt, wenn tatsächlich Fehler oder Warnungen erkannt wurden. Fehlt die Datei, ist die Verarbeitung fehlerfrei verlaufen.

Beispiele aus Testdaten

Im Ordner [testdata/](#) befinden sich mehrere Beispieldateien, die gezielt Fehlerfälle demonstrieren:

DXF-Datei	Fehlertyp	Beschreibung
easy_tunnelerror1.dwg	<code>overdefined_tunnel</code> , <code>missing_attributes</code>	TUNNEL2 hat 3 Positionen statt 2; Block BX0101 hat fehlende Attribute
easy_3tunnels.dxf	Diverse Tunnel-Fehler	Demonstriert verschiedene Tunnel-Konfigurationen

Diese Testdateien können als Referenz dienen, um Fehler im eigenen Layout zu verstehen und zu beheben.

Anpassung des Verhaltens des Programms:

Verwendete Konfigurationsdateien

Zum aktuellen Zeitpunkt verwendet bzw. berücksichtigt das Programm **vier** Konfigurationsdateien (Dateiendung `.cfg`). Diese steuern das Verhalten des Programms und **können vom Nutzer bei Bedarf geändert** werden bzw. **müssen vom Nutzer im Falle von betrieblichen Änderungen aktualisiert werden** (z.B Änderung von Sachnummern), um die Funktion des Programms zu gewährleisten. In der folgenden Tabelle sind die Aufgaben bzw. Inhalte der einzelnen Konfigurationsdateien aufgezeigt und in den darauffolgenden Abschnitten der jeweilige Aufbau und Besonderheiten erläutert.

[!IMPORTANT] Die Einträge in den `.cfg`-Dateien müssen **laufend gepflegt** werden, wenn sich:

- Layernamen in CAD-Dateien ändern
- Neue BMK-Kürzel eingeführt werden
- Neue SIVAS-Artikel hinzukommen

Nur so kann eine korrekte automatische Kabelberechnung und Zuordnung gewährleistet werden.

Datei	Inhalt
<code>allgemein.cfg</code>	Layernamen der Kabelpritschen Layernamen der Unterverteiler Layernamen der Tunnel Layernamen der Sensoren / Aktoren Toleranz der Verbindung zw. Sensor und Kabeltrasse Toleranz des "Anpinnens" zweier Kabeltrassen untereinander

Datei	Inhalt
BMK.cfg	Betriebsmittelkennzeichnung der Elemente, die im Routing berücksichtigt bzw. ignoriert werden sollen Zuweisung der Kabelkennzeichnungen zur jeweiligen Betriebsmittelkennzeichnung Längen Anpassungen für einzelne Kabelkennzeichnungen
kabel.cfg	SIVAS-Nummern für die einzelnen längenabhängigen Kabel der jeweiligen Kabelkennzeichnung
bezeichner.cfg	Speicherung der Bezeichner zu den SIVAS-Nummern sowie automatische Pflege fehlender Artikelnummern

[!IMPORTANT]

Änderungen an den .cfg-Dateien werden erst beim nächsten Programmlauf wirksam. Stelle sicher, dass die Datei korrekt gespeichert wurde und keine doppelten Abschnitte enthält.

Zusammenfassung und Struktur der Konfigurationsdateien

Nachfolgende Aufzählung zeigt den Aufbau und Zweck der .cfg-Dateien für das Routing-Tool. Jeder Abschnitt ist dokumentiert mit:

- Konfigurations-Block (Abschnittsname)
- Beschreibung der Funktion
- Beispielhafte Einträge

allgemein.cfg

Diese Datei steuert das Einlesen von Layern und definiert geometrische Toleranzen.

Abschnitt	Beschreibung	Beispiel(e)
[GetPos-Layer_Racks]	Layer mit Kabelpritschen (Racks)	PRITSCH_200-60_OMNIFLO 0-0_ILS_Pritsche_200-60
[GetPos-Layer_Distributors]	Layer mit Unterverteilern	0-0_ILS_UNTERVERTEILER UNTERVERTEILER
[GetPos-Layer_Tunnel]	Layer für Tunnel	Busverteiler-Kennzeichnung
[GetPos-Layer_Equipment]	Layer mit Sensoren, Motoren, IOs etc.	0-0_ILS_MOTOR MOTOR REALE_POSITION_IO
[GetPos-Geom-Sensor]	Maße von Sensor-Markern zur Mittelpunktberechnung	Breite = 80 Höhe = 90
[Racks]	Snap-Toleranz zwischen benachbarten Racks	SnapTolerances = 200

Abschnitt	Beschreibung	Beispiel(e)
[Sensoren]	Verbindungstoleranz Sensoren ↔ Racks	ConnectionTolerances = 3000

BMK.cfg

Diese Datei steuert, **welche Betriebsmittelkennzeichen (BMK)** beim Routing berücksichtigt werden und welche **Kabeltypen** zugewiesen sind.

Abschnitt	Beschreibung	Beispiel(e)
[Routing-Include]	BMKs, die beim Routing berücksichtigt werden	MA = Motoren QM = Ventile BX = Scanner
[Routing-Ignore]	BMKs, die ignoriert werden	FC = Frequenzumrichter QA = Sicherungen
[Cable-Mapping]	Zuordnung BMK → Kabelgruppe(n) (aus <code>kabel.cfg</code>)	MA → MA MB → WD_Q BX → WF_B, WD_I
[Length-Adjustments]	Kabellängen-Korrektur (z. B. vorkonf. Sensorleitung)	BX → 4 m abziehen

kabel.cfg

Diese Datei enthält die **SIVAS-Nummern je Kabellänge und -typ** sowie einen Abschnitt für **spezifische Kabelgruppen**

Abschnitt / Gruppe	Beschreibung	Beispiel(e)
[WD_Q], [WD_I], [WF_B]	Kabellängen je Gruppe + zugehörige SIVAS-Nr.	1.0 m → 722001300 2.5 m → 722001338 5.0 m → 726001045
sensorspezifische Gruppen:		
[WD_I-829422026]	Kabellisten für bestimmte Artikelnummern	5.0 m → 722001353
[WD_I-720002003]		10.0 m → 722001354

bezeichner.cfg

Diese Datei enthält Bezeichner zu den verwendeten SIVAS-Artikelnummern. Sie beinhaltet sowohl die Bezeichner (laut SIVAS) für die Bauteil-Artikelnummern als auch für die Kabel. Die Datei wird im Laufe eines Routing-Prozesses bei Bedarf angepasst und mittels einer SIVAS-Datenbankabfrage (bei bestehender VPN-Verbindung bzw. im lokalen Firmennetz am Standort) erweitert.

Abschnitt	Beschreibung	Beispiel(e)
-----------	--------------	-------------

Abschnitt	Beschreibung	Beispiel(e)
[Sivasnummern]	Bekannte SIVAS-Nr. mit zugehörigem Bezeichner	725000015 → 4G1,5mm ² Steuerleitung 722001300 → Verb.-Leitung M12 St/M8 Bu 1 m
[Missing]	Fehlende Nummern, die im Layout vorkommen, aber unbenannt sind	Automatisch ergänzt bei VPN/SIVAS-Datenbankzugriff

Validierung der Konfigurationsdateien

Bevor das Routing beginnt, werden alle **.cfg**-Dateien automatisch auf **Plausibilität und Konsistenz** überprüft. Ziel ist es, fehlerhafte oder unvollständige Konfigurationen frühzeitig zu erkennen und entsprechende Warnungen auszugeben.

Diese Prüfung geschieht vollautomatisch im Hintergrund – der Anwender muss nichts weiter tun. Falls Probleme erkannt werden, werden diese in der Konsole (bzw. dem Terminal-Fenster) protokolliert. Die Verarbeitung wird trotzdem fortgesetzt, allerdings **können fehlerhafte Konfigurationen zu falschen Verkabelungen oder fehlenden Ausgaben führen**.

Prüfkriterien im Detail:

1. Vollständigkeit der Zuweisungen in **BMK.cfg**

- Jeder Prefix aus [Routing-Include] muss in [Cable-Mapping] enthalten sein.
- Fehlt eine Zuordnung, wird ein Fehler gemeldet:
`No Cable-Mapping for Prefix 'BX' within [Cable-Mapping]`

2. Existenz der referenzierten Sektionen in **kabel.cfg**

- Jeder Eintrag in [Cable-Mapping] verweist auf eine oder mehrere Sektionen (z. B. **WD_Q**, **WF_B**), die in **kabel.cfg** existieren müssen.
- Fehlt eine Sektion, wird gewarnt:
`Cable-Section 'WF_B' from Cable-Mapping is not defined in kabel.cfg`

3. Gültigkeit der Werte in [Length-Adjustments]

- Jeder Wert muss eine **nicht-negative Kommazahl (float ≥ 0.0)** sein.
- Ungültige oder negative Einträge verursachen entsprechende Warnungen: `Invalid Value in Length-Adjustments for BX: 'abc' is not a valid float`

[!NOTE] Die Validierung dient als wichtige Schutzmaßnahme gegen fehlerhafte Eingaben oder veraltete Konfigurationsdateien. Es ist **empfohlen**, bei jeder Layoutanpassung oder Änderung von Komponenten auch die **.cfg**-Dateien zu prüfen und ggf. zu aktualisieren, um die Richtigkeit der Ergebnisse sicherzustellen.

Anwenderhinweise zur Layouterstellung

Im nachfolgenden Abschnitt sind die **wichtigsten** Punkte zur Erstellung der Layoutzeichnungen aufgeführt. Die Beachtung dieser Hinweise stellt sicher, dass die automatische Auswertung durch das Programm fehlerfrei und zuverlässig erfolgen kann.

[!IMPORTANT] **Dieser Abschnitt ist durch die Fachabteilung regelmäßig zu pflegen und analog zu Konfigurationsdateien stets aktuell zu halten.**

Verwendung der Layer

- **Erfasst werden ausschließlich Elemente auf Layern**, die in der Konfigurationsdatei (**cfg**) unter dem jeweiligen Abschnitt (**GetPos-Layer_...**) aufgeführt sind.
- Nicht konfigurierte Layer oder falsch benannte Layer werden vollständig ignoriert.
- Für jede Objektklasse (z. B. Racks, Sensoren, Unterverteiler) sollte ein eigener dedizierter Layer verwendet werden.
- Layer-Namen dürfen **keine Sonderzeichen oder Leerzeichen** enthalten, und sollten **eindeutig** benannt sein.

Zeichnen von Kabeltrassen

- **Verbindungspunkte von Racks sollten präzise aneinander gezeichnet werden**, idealerweise mit Fangmodi (Snapping) im CAD.
 - Das Programm erkennt und verbindet nahe beieinanderliegende Racks automatisch, jedoch nur mit begrenzter Toleranz.
 - Eine saubere Planung der Trassengeometrie erleichtert die automatische Graphenerstellung erheblich.
- **2D-Kabeltrassen (LWPOLYLINE)** sollten verwendet werden für Strecken in einer Ebene bzw. auf einer Höhe
- **3D-Trassenführung** bei Höhendifferenzen (z.B entlang von Förderern) erfolgt in drei Schritten:
 1. Untere Ebene als **LWPOLYLINE** mit Z=0.
 2. Obere Ebene als **LWPOLYLINE** mit Z=Erhebung (über DXF-Attribut **Erhebung**).
 3. Verbindung über **3DPOLYLINE**, welche die Steigung darstellt.

Attribute der Blöcke (Bauteile)

- **Pro Bauteil darf nur ein Block vorhanden sein.** Alle relevanten Informationen (z. B. Artikelnummer, Bezeichner, Position) müssen diesem einen Block als Attribute zugeordnet sein.
- **Keine übereinanderliegenden Blöcke!**
 - In der Vergangenheit wurden häufig zwei Blöcke übereinandergelegt, um Kabel- und Sensorinformationen zu trennen. Dies ist künftig nicht mehr zulässig.
- **Beispiel für frühere Aufteilung (nicht mehr erlaubt):**
 - *Rahmen innen* (Angaben zum Kabel):
 - A:
 - B: Kabel-ID
 - C: lange Nummer (?)
 - ArtiNr: Kabelartikelnummer
 - Beschr: Kabelname (SIVAS)
 - Menge, Position, Gruppe, Etiketle, Auflösung, etc.
 - *Rahmen außen* (Angaben zum Bauteil selbst):

- A:
- B: Bauteil-Kurzbezeichnung
- C: lange Nummer (?)
- ArtiNr: Sensorartikelnummer
- Beschr: Kurzbeschreibung
- sowie gleiche Attribute wie oben.
- *Text*
 - IO: Bauteil-Kurzbezeichnung
 - id
 - VERW
 - BEZEICHNUNG
 - TEXT-D, TEXT-E, TEXT-ES, TEXT-F
 - SPS
- **In Zukunft:**
 - Ein einziger Block enthält alle Informationen des Bauteils.
 - Die Informationen zu den benötigten Kabeln werden **allein** vom Kabeltool bereitgestellt
 - Die konkrete Struktur der Attribute wird noch festgelegt (To Be Defined).
 - Ziel ist es, sowohl die Verarbeitung im System als auch die manuelle Pflege im Layout zu vereinfachen.

Positionierung von Bauteilen

- In Zukunft wird das Attribut **REALE_POSITION** eingeführt. Damit kann die **exakte physische Einbauposition** eines Bauteils (z. B. Sensors) unabhängig von der tatsächlichen Platzierung des Blocks im Layout definiert werden.
 - Ermöglicht eine saubere, leserliche Platzierung der Blöcke im Plan – z. B. seitlich der Anlage (dort wo Platz ist)
 - Der Block selbst kann damit **optisch gut positioniert**, aber technisch korrekt ausgewertet werden.
- Bauteile ohne **REALE_POSITION** werden standardmäßig an der **gezeichneten Blockposition** verortet.
- **Anbindung an Kabeltrassen erfolgt immer zur nächstgelegenen Rack-Geometrie** im definierten Toleranzbereich.
 - Wird ein Block zu weit entfernt vom Rack gezeichnet (bzw. **REALE_POSITION** liegt zu weit weg), erfolgt **keine automatische Verbindung**.
 - Die maximale Toleranz ist über die Konfiguration **allgemein.cfg[tol_connect]** steuerbar.
- Es ist daher darauf zu achten, dass Bauteile (bzw. deren reale Positionen) **räumlich korrekt und in sinnvoller Nähe** zu den Kabeltrassen platziert sind.